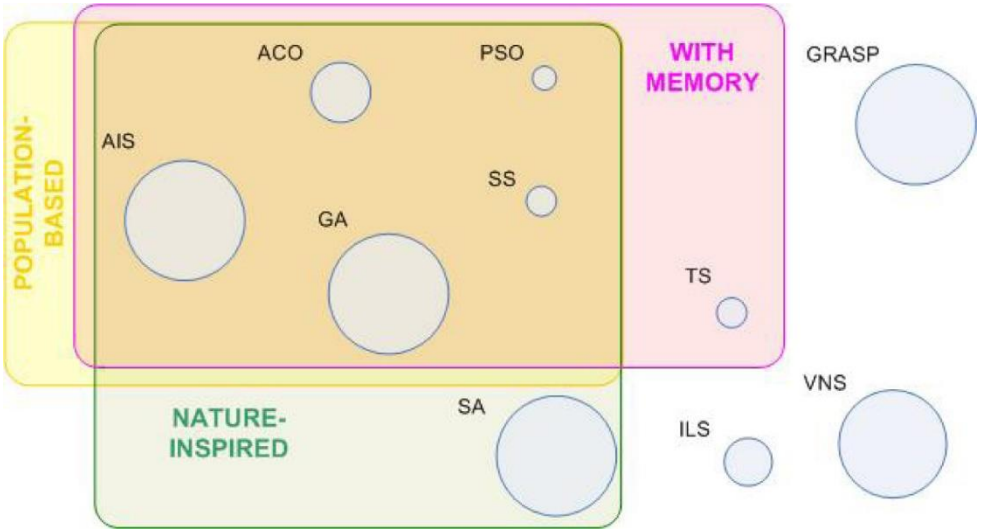
Algoritmos Avanzados

Metaheurísticas

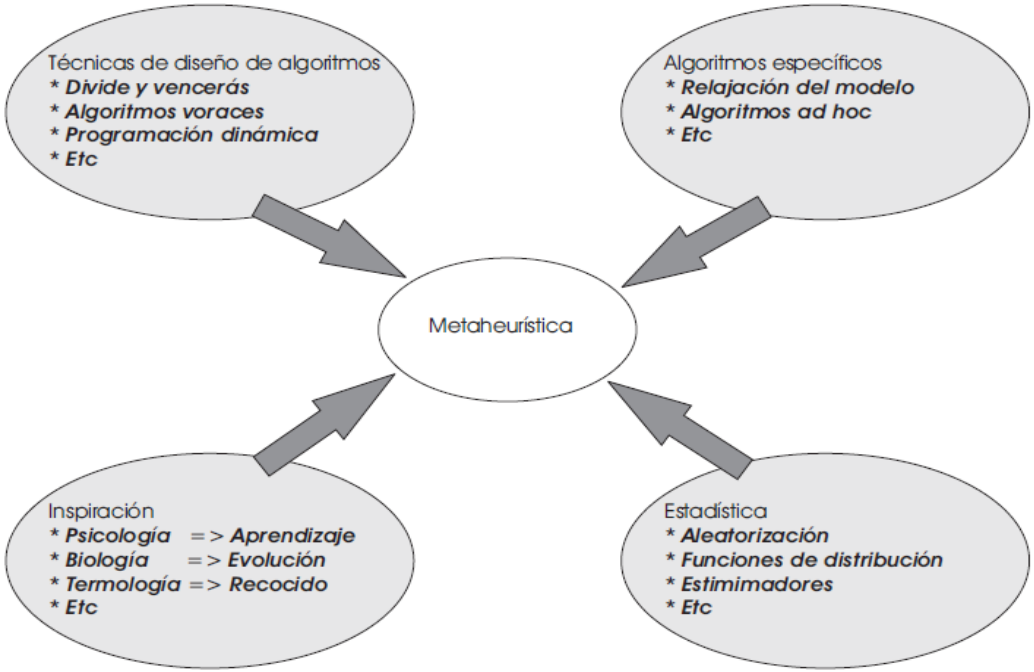
Metaheurísticas

El término Metaheurística o Meta-heurística fue acuñado por F. Glover en el año 1986 [121]. Este término deriva de la composición de dos palabras con origen griego, que son ”meta” y ”heurística”. El segundo término ha sido descrito en el capítulo anterior, mientras que el prefijo meta (en inglés) se podría traducir como **más de un nivel superior.**



Metaheurísticas

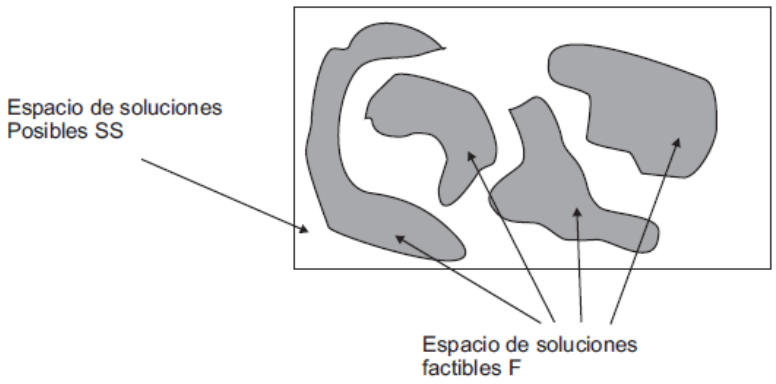
Se puede decir que las metaheurísticas combinan ideas que provienen de cuatro campos de investigación distintos: las técnicas de diseño de algoritmos (resuelven una colección de problemas), algoritmos específicos (dependientes del problema que se quiere resolver), fuente de inspiración (del mundo real) y métodos estadísticos.



Metaheurísticas

Existen tres conceptos básicos que se pueden encontrar en la resolución algorítmica de, prácticamente, cualquier problema de optimización combinatoria. Independientemente de la técnica que se utilice, se debe especificar:

• **Representación**: se encarga de codificar las soluciones factibles para su manipulación. Determina el tamaño (cardinalidad) del espacio de búsqueda (SS) de cada problema.



Metaheurísticas

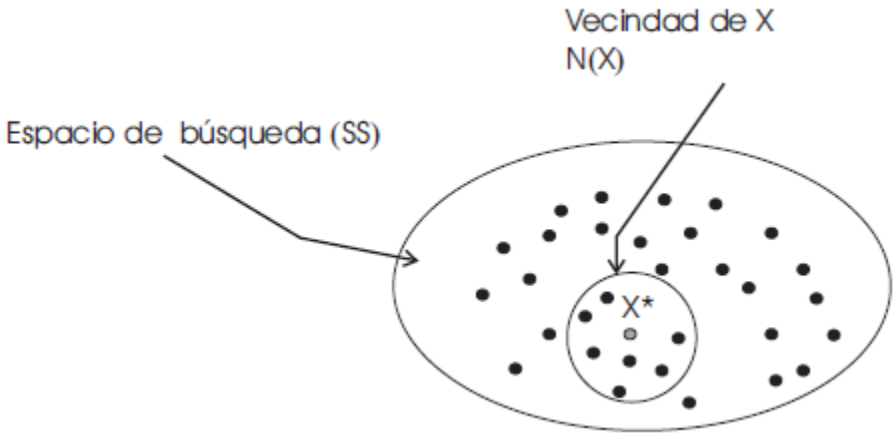
• **Objetivo:** describe el propósito que se debe alcanzar para una representación dada. Es un predicado matemático que expresa la tarea que se tiene que realizar.

• **Función de evaluación:** permite asociar a cada solución factible un valor que determina su calidad. Es una correspondencia f entre los puntos del espacio de soluciones y R.

Cuando se diseña una función de evaluación se debe tener en cuenta que para muchos problemas no es suficiente con encontrar una solución, sino que ésta debe ser una solución factible; es decir, que satisfaga una serie de restricciones impuestas por el problema. Por lo tanto, se debe establecer una diferencia entre el espacio de búsqueda SS y el espacio de soluciones factibles $F \subseteq SS$.

Metaheurísticas (términos)

- **Vecindad:** Dada una solución x ∈ SS, la vecindad N(x) de esa solución es un subconjunto del espacio de soluciones que contiene soluciones que están "**próximas**" de la solución considerada. En la figura se muestra una representación gráfica de la vecindad de una determinada solución x.

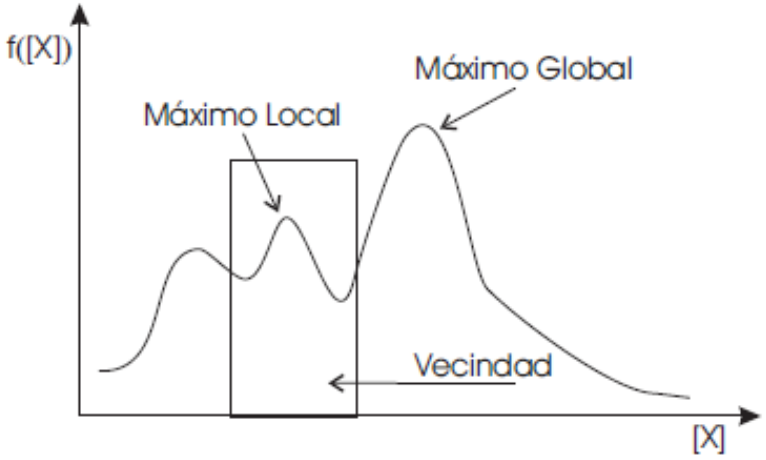


Metaheurísticas (términos)

• **Óptimo local:** Dado un problema de optimización P = (f ,SS,F) y una estructura de vecindad N, se dice que una solución factible x ∈ F ⊆ SS es un óptimo (máximo) local con respecto a N si:

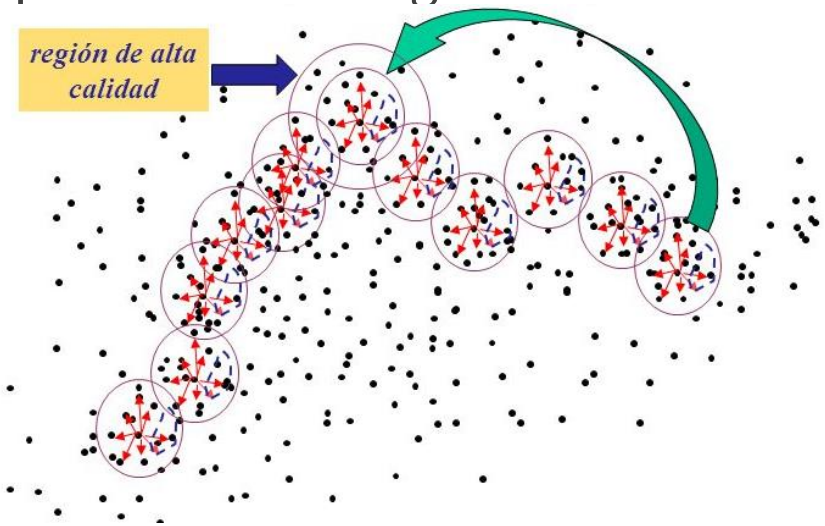
$$\forall y \in N(x)\ f(x) \geq f(y)$$

En la se representan gráficamente el máximo global, el máximo local y la vecindad, para una función objetivo dada.



Metaheurísticas (términos)

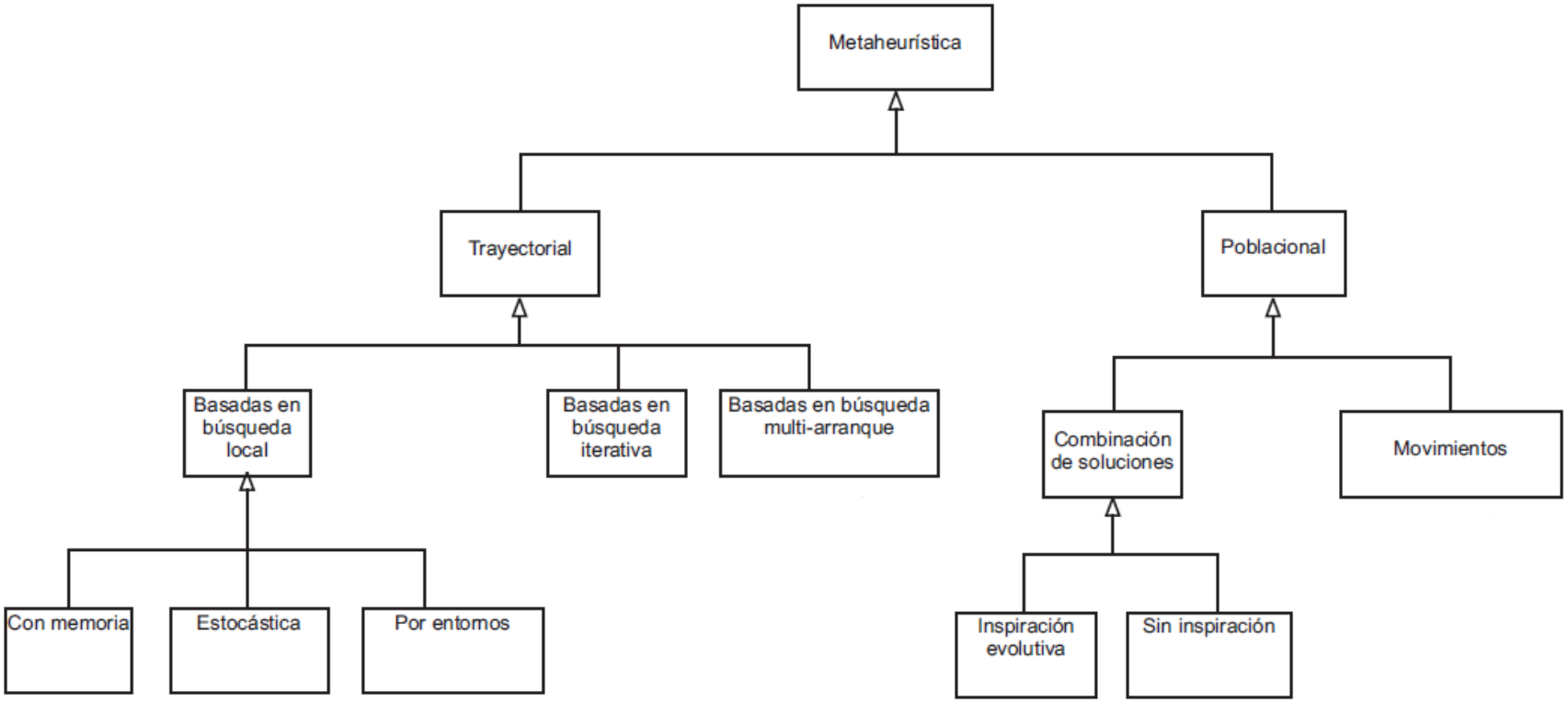
• **Intensificación:** es la forma de definir el proceso de búsqueda más exhaustivo que puede llevar a cabo una metaheurística en una vecindad dada. Generalmente, la metaheurística no intensifica en cualquier entorno, ya que eso sería una búsqueda exhaustiva de todo el espacio de soluciones, sino que tiene en cuenta factores como, por ejemplo, la calidad de las soluciones encontradas en esa vecindad. Es decir, si en esa vecindad no hay ninguna solución de calidad no parece interesante intensificar la búsqueda en dicha región.



Metaheurísticas (términos)

• **Diversificación:** es la forma de definir el proceso mediante el cual la metaheurística es capaz de visitar diversas vecindades lejanas. Habitualmente, la metaheurística no diversifica constantemente y sin criterio, ya que eso sería una búsqueda aleatoria en el espacio de soluciones, sino que tiene en cuenta factores, como por ejemplo el hecho de que una región no haya sido visitada.

Clasificación jerárquica de las metaheurísticas



Clasificación de Metaheurísticas

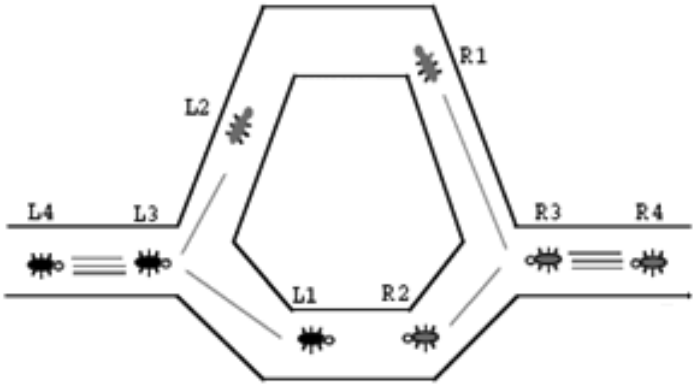
• **Trayectoriales:** estos métodos se caracterizan por una trayectoria en el espacio de soluciones. Es decir, que partiendo de una solución inicial, son capaces de generar un camino o trayectoria en el espacio de búsqueda a través de operaciones de movimiento. Las características de la trayectoria proporcionan información sobre el comportamiento del algoritmo así como de su efectividad. En esta clase tenemos por ejemplo:

- Búsqueda tabú
- Recocido simulado
- Búsqueda de vecindad variable
- Búsqueda local guiada
- GRASP
- Búsqueda por vecindades adaptativa y borrosa (FANS)

Clasificación de Metaheurísticas

• **Poblacional:** son aquéllas que emplean un conjunto de soluciones (población) en cada iteración del algoritmo, en lugar de utilizar una única solución como las metaheurísticas trayectoriales. Estas metaheurísticas proporcionan de forma intrínseca un mecanismo de exploración paralelo del espacio de soluciones, y su eficiencia depende en gran medida de cómo se manipule dicha población. En esta clase tenemos por ejemplo:

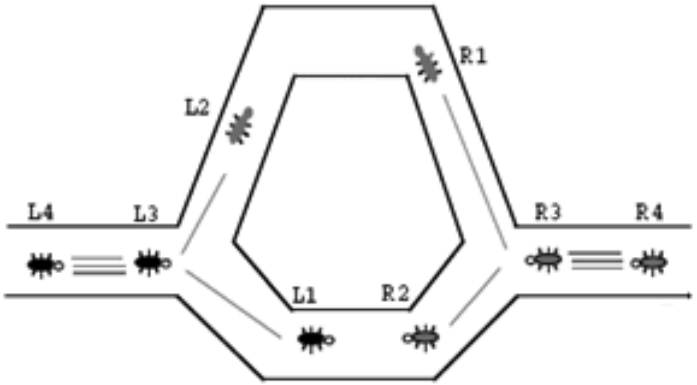
- Algoritmo Genético (AE)
- Algoritmo Memético (AE)
- Búsqueda dispersa
- Algoritmo Cultural
- Optimización por colonias de hormigas (ACO)
- Optimización por enjambre de partículas (PSO)



Clasificación de Metaheurísticas

• **Poblacional:** son aquéllas que emplean un conjunto de soluciones (población) en cada iteración del algoritmo, en lugar de utilizar una única solución como las metaheurísticas trayectoriales. Estas metaheurísticas proporcionan de forma intrínseca un mecanismo de exploración paralelo del espacio de soluciones, y su eficiencia depende en gran medida de cómo se manipule dicha población. En esta clase tenemos por ejemplo:

- Algoritmo Genético (AE)
- Algoritmo Memético (AE)
- Búsqueda dispersa
- Algoritmo Cultural
- Optimización por colonias de hormigas (ACO)
- Optimización por enjambre de partículas (PSO)



GRASP (Greedy Randomized Adaptative Search Procedure)

Esta técnica fue desarrollada por T. Feo y M. Resende a finales de los años 80. Mientras que el criterio voraz permitía seleccionar solamente el mejor valor de la función objetivo a optimizar, los algoritmos GRASP relajan o amplían este criterio de tal manera que, en vez de seleccionar un único elemento, forma un conjunto de elementos candidatos a ser parte del conjunto solución y que cumplen ciertas condiciones.

Es sobre este conjunto formado que realizará una selección aleatoria de algún elemento.

GRASP (Greedy Randomized Adaptative Search Procedure)

Los GRASP tienen las siguientes características:

• Son procedimientos de **búsqueda**: dentro de un espacio de posibles soluciones, se ejecutan búsquedas sin evaluar a todos los elementos del problema.

• Son **voraces**: porque en cada evaluación escoge a los mejores candidatos que cumplan ciertas condiciones.

• Son **adaptables**: porque se adapta a la estructura de la instancia del problema que pretende resolver.

• Son **aleatorias**: porque de entre las candidatas escoge aproximadamente al azar, aquellas que finalmente formarán parte de la solución relajando el criterio voraz.

GRASP (Greedy Randomized Adaptative Search Procedure)

Procedimiento GRASP General (Instancia del problema)

Inicio

1. Leer Instancia
2. Mientras <no se cumpla condición de parada> hacer
    - 2.1 Procedimiento construcción (S_k)
    - 2.2 Procedimiento Mejoría (S_k)

    Fin Mientras
3. Retornar (Mejor S_k)

Fin GRASP

GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de construcción

Esta etapa consiste en construir una solución voraz relajada o ampliada en donde exista más de una candidata a ser solución y a partir de la cual, se seleccionará aleatoriamente elementos para formar la solución final. Recuérdese que el criterio voraz que buscaba optimizar la función objetivo puede ser planteado de la siguiente forma: Mejor $\{c(x) / x \epsilon N\}$ en donde c es una función voraz objetivo a optimizar. Aquí el criterio GRASP modifica al criterio voraz ampliándolo, de forma tal que se pueda construir una lista de varios elementos candidatos a ser solución. Primero determina los mejores y peores valores de la función c para luego formar la lista de candidatas.

$$\beta = \text{Mejor } \{c(x) : x \epsilon N\}$$

$$\tau = \text{Peor } \{c(x) : x \epsilon N\}$$

GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de construcción

Luego se establece el conjunto de candidatas RCL (por sus siglas del inglés: Restricted Candidates List) que se formará con todos los elementos x de N que cumplan con la condición:

$$RCL = \{ x \in N : \beta \leq c(x) \leq \beta + \alpha (\tau - \beta) \}$$

o

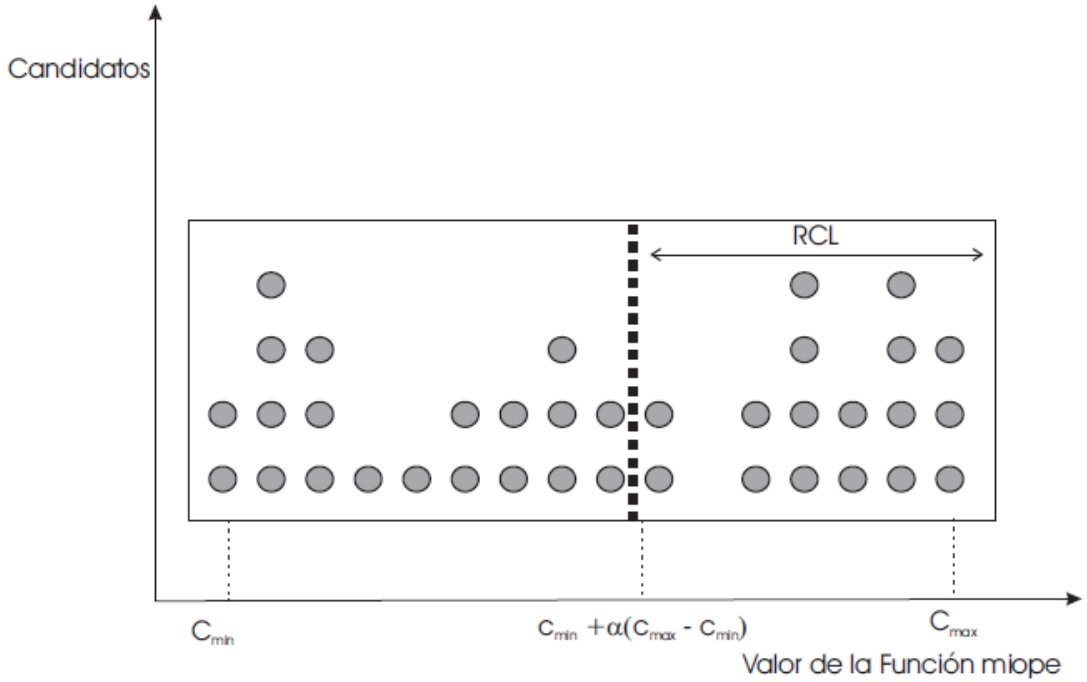
$$RCL = \{ x \in N : \beta - \alpha (\beta - \tau) \leq c(x) \leq \beta \}$$

Al parámetro α que aparece en la definición de RCL se le conoce como parámetro de relajación GRASP y es el responsable de tornar menos voraz a la solución que surja. Del conjunto RCL se toma un elemento al azar que pasa a formar parte del conjunto S, repitiéndose el proceso hasta que se hayan procesado todos los elementos de N.

GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de construcción

Lista de candidatos restringida para problemas de maximización:



GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de construcción

Una estrategia aleatoria viene a ser particularmente útil cuando existen varias formas para realizar un paso dentro de un algoritmo en general y donde resulta difícil garantizar totalmente que alguna de ellas es la mejor elección.

La calidad de la solución va a depender directamente del valor del α empleado. Según el valor que tome α.

$\alpha = 1 \rightarrow$ Criterio totalmente aleatorio

$\alpha = 0 \rightarrow$ Criterio totalmente voraz

$0 \leq \alpha \leq 1 \rightarrow$ Criterio convencional que requiere un proceso de calibración adaptado a cada problema.

GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de construcción

Las estructuras de datos que van a emplearse en el algoritmo han sido mencionadas anteriormente:

N: Conjunto de datos de donde se obtendrán las variables del problema
x: Elemento del conjunto N
S: Conjunto solución
c: Función de voraz de mérito a optimizar

GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de construcción

La siguiente figura muestra al algoritmo detallado de la fase de construcción:

Procedimento GRASP Construcción(N, c, α, S)
1. Inicializar N
2. S= ϕ
3. Mientras <no se cumpla condición de parada> hacer
Inicio
3.1 RCL = ϕ
3.2 β = Mejor {$\forall$ x$\in$ N / c(x)}
3.3 τ = Peor {$\forall$ x$\in$ N / c(x)}
3.4 RCL = {$\forall$ x$\in$ N / $\beta \leq$ c(x) $\leq \beta + \alpha^*(\tau - \beta)$
3.5 a = Aleatorio (RCL)
3.6 Si S $\cup$ {a} es una solución viable al problema entonces S = S $\cup$ {a}
3.7 N = N – {a}
Fin Mientras
4. Retornar S
Fin GRASP Construcción.

GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de mejoría

Como en los casos de las heurísticas, las soluciones construidas por medio de algoritmos GRASP no son necesariamente óptimas locales, por lo que se hace necesaria la aplicación de un procedimiento de mejora de la solución encontrada en la fase de construcción.

Siendo la GRASP una generadora de buenas soluciones, se espera que al menos una de ellas este dentro del entorno o vecindad F de la solución óptima. Lo primero que establece esta fase de mejoría entonces, es la definición de una vecindad de soluciones alrededor de la solución hallada en la fase de construcción.

GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de mejoría

Esta vecindad puede ser determinada intercambiando uno a uno los elementos de la solución con aquellos que no pertenecen a la solución, siempre que se verifique la propiedad F. La forma más sencilla de trabajo es limitar la búsqueda a k-vecindades. Dado un vector binario X= {x_1, x_2, x_3, x_4…. x_N} y un entero positivo k, se puede definir el conjunto siguiente:

$$N_k(X) = \{z_1, z_2, z_3, \ldots z_N \} tq \sum_j^N |z_j - x_j| \leq k, z_j \in \{0,1\}$$

Este conjunto es denominado k-vecindad de X e incluye los elementos que son distintos de X en no más de k componentes. Un intercambio con k permutaciones, permitiría mover un punto i-ésimo de X hacia otro punto i+1-ésimo hasta que no se produzcan mejoras en las soluciones que aporta X.

GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de mejoría

Procedimiento GRASP Mejoría (X)
Mientras (es posible mejorar solución X) hacer
Inicio

1. $N_k(X) = \{z_1, z_2, z_3, \ldots z_N \} tq \sum_j^N |z_j - x_j| \leq k, z_j \in \{0,1\}$
2. $Best_k(N_k) = \{C_i^{i+1} z_i\}$
3. **Si** $Best_k(N_k)$ **es mejor que X** $\Rightarrow$ **X =** $Best_k(N_k)$

Fin Mientras
Retornar(X)
Fin GRASP Mejoría

GRASP (Greedy Randomized Adaptative Search Procedure)

Fase de mejoría

La mejoría GRASP es considerada entonces una fase de post procesamiento y su aplicabilidad depende de las restricciones del problema y de los niveles de calidad requeridos para solucionarlo. Los algoritmos de refinamiento son complejos computacionalmente y a la vez demandan altos recursos de hardware.

Esta fase de búsqueda local es iterativa y va sucesivamente reemplazando la solución actual por otra mejor que esté en el vecindario, terminando cuando no encuentra una mejor solución que pueda servir de reemplazo. Dos son los enfoques básicos de mejoramiento:

- **Primer movimiento**: todos los miembros de la vecindad formada son analizados y la solución actual es reemplazada por la solución del mejor vecino identificado.

- **Mejor movimiento**: la solución actual es reemplazada por la solución del primer miembro de la vecindad cuyo valor sea mejor.

Bibliografía Recomendada

Duarte, Pantrigo y Gallego. **Metaheurísticas**. Madrid. Universidad Rey Juan Carlos, 2007.

TUPIA, M. **Fundamentos de Inteligencia Artificial**. 2da edición. Perú: 2014